April 1, 2002

Dear Mr. Pannala,

Attached please find the results of your search request for application #09/512,949. I looked for combinations of nearest neighbor searches with both local polar coordinates and, separately, partitioned cells of a dataspace. Within those sets, I also looked for the other concepts as I felt they should already be there. I searched Dialog and IBM's TDBs.

Please let me know if you have any questions.

Regards,

Geoffrey St. Leger
4B30/308-7800

An Efficient Indexing Method for Nearest Neighbor Searches in High-Dimensional Image Databases

Guang-Ho Cha, Xiaoming Zhu, Dragutin Petkovic, *Fellow, IEEE*, and Chin-Wan Chung

***Abstract*—Nearest neighbor (NN) search is emerging as an important search paradigm in a variety of applications in which objects are represented as vectors of d numeric features. However, despite decades of efforts, except for the *filtering approach* such as the VA-file [31], the current solutions to find *exact* kNNs are far from satisfactory for large d. The filtering approach represents vectors as compact approximations and by first scanning these smaller approximations, only a small fraction of the real vectors are visited. In this paper, we introduce the *local polar coordinate file* (*LPC-file*) using the filtering approach for nearest-neighbor searches in high-dimensional image databases. The basic idea is to partition the vector space into rectangular cells and then to approximate vectors by *polar coordinates* on the partitioned local cells. The LPC information significantly enhances the discriminatory power of the approximation. To demonstrate the effectiveness of the LPC-file, we conducted extensive experiments and compared the performance with the VA-file and the sequential scan by using synthetic and real data sets. The experimental results demonstrate that the LPC-file outperforms both of the VA-file and the sequential scan in total elapsed time and in the number of disk accesses and that the LPC-file is robust in both "good" distributions (such as random) and "bad" distributions (such as skewed and clustered).**

***Index Terms*—Dimensionality curse, image database, indexing method, nearest neighbor (NN) search.**

I. Introduction

IMAGE databases often represent the image objects as vectors of d numeric features and access them via the feature vectors and similarity measures. Feature vectors can be viewed as points in a d-dimensional vector space and the similarity measure can be viewed as a measure of distance within that space. An important problem in such a system is to find the k most similar images to a given sample image. This problem is typically solved by first mapping the sample image to its corresponding feature vector and then finding the kNNs of a vector. The basic issue then is how to find the kNNs efficiently.

Manuscript received May 22, 2000; revised February 27, 2001. This work was supported by the Basic Research Program of the Korea Science and Engineering Foundation under Grant 2000-1-51200-007-3. The associate editor coordinating the review of this paper and approving it for publication was Dr. Anna Hac.

G.-H. Cha is with the Department of Multimedia Engineering, Tongmyong University of Information Technology, Pusan 608-711, South Korea (e-mail: ghcha@tmic.tit.ac.kr).

X. Zhu was with IBM Almaden Research Center, San Jose, CA 95120 USA. He is now with eLance.com, Sunnyvale, CA 94086 USA (e-mail: xzhu@elance.com).

D. Petkovic was with the IBM Almaden Research Center, San Jose, CA 95120 USA. He is now with VMware, Palo Alto, CA 94304 USA (e-mail: dragutin@vmware.com).

C.-W. Chung is with the Department of Computer Science, Korea Advanced Institute of Science and Technology, Taejon 305-701, South Korea (e-mail: chungcw@islab.kaist.ac.kr).

Publisher Item Identifier S 1520-9210(02)01399-8.

A. Motivation

For applications where the vectors have low or medium dimensionalities (e.g., less than ten), existing multidimensional indexing methods (MIMs), such as the R^*-tree [3], the X-tree [5], the HG-tree [6], the LSD^h-tree [15], and the SR-tree [19], can be usefully employed to solve the problem. So far, however, there is no effective solution to this problem for the applications in which the vectors have high dimensionalities, e.g., over 100. In fact, for a high dimensionality, in theory or in practice, the performance of existing MIMs degenerates to being worse than that of the brute-force sequential scan that compares the query object to each data object [17], [31]. Therefore, the main issue is to overcome the *curse of dimensionality* [25]—a phenomenon that the performance of indexing methods degrades drastically as the dimensionality increases. With this urgent need, we aim at developing a new indexing method for k-NN searches that significantly improves the performance compared with the sequential scan.

B. The k-Nearest Neighbor (kNN) Problem

Consider a database DB consisting of points from $S = R_1 \times \cdots \times R_d$, where $R_i \subseteq \Re$. Each R_i usually consists of either integers or floats. A k-NN query consists of a point $\boldsymbol{q} \in S$ and a positive integer k. The k-NN search finds the nearest k neighbors of $\boldsymbol{q}$ with respect to a distance function $\|\cdot\|$. The output set O consists of k points from the database such that

$$\forall \boldsymbol{a} \in 0 \text{ and } \forall \boldsymbol{b} \in DB - 0 \quad \|\boldsymbol{q} - \boldsymbol{a}\| \leq \|\boldsymbol{q} - \boldsymbol{b}\|.$$

The Euclidean distance between a pair of objects is commonly used as a dissimilarity measure. The typical way to compute the dissimilarity between two objects is using an L_p distance metric. Let $\boldsymbol{p} = (p_1, p_2, \ldots, p_d)$ and $\boldsymbol{q} = (q_1, q_2, \ldots, q_d)$ be two points in the d-dimensional space. Then the distance $\|p - q\|$ between $\boldsymbol{p}$ and $\boldsymbol{q}$ using an L_p metric is as follows:

$$\|\boldsymbol{p} - \boldsymbol{q}\| = L_p(\boldsymbol{p}, \boldsymbol{q}) = \left(\sum_{i=1}^{d} |p_i - q_i|^p \right)^{1/p}$$

where p is the order of the metric. For $p = 1$ and $p = 2$, we obtain the well-known Manhattan and the Euclidean distance, respectively. The actual problem in database applications is how to process such queries so that the nearest k objects can be returned within a desired response time. Therefore, our focus is the development of indexing method to accelerate the speed of the k-NN search. We provide a new indexing method called the *local polar coordinate file (LPC-file)* that significantly improves the k-NN search speed in high-dimensional data spaces.

C. Similarity Measure, Assumptions and Notations

The test bed for our exploration of the k-NN problem is IBM's query by image content (QBIC) [13], [25]. We focus on the specific problem of retrieving images via a 256-dimensional color histogram, using a QBIC's special-purpose color-similarity measure. To retrieve relevant images, QBIC uses color matching based on the distribution of colors that occur in an image, not on the average color. In this way, a red and blue object matches other red and blue objects better than to purple objects. To match color histograms, the distance $\|\boldsymbol{p} - \boldsymbol{q}\|$ of the color histogram $\boldsymbol{p}$ for each image to the histogram $\boldsymbol{q}$ of the query image is computed as

$$\|\boldsymbol{p} - \boldsymbol{q}\| = (\boldsymbol{p} - \boldsymbol{q})^T \boldsymbol{A} (\boldsymbol{p} - \boldsymbol{q}).$$

The histograms $\boldsymbol{p}$ and $\boldsymbol{q}$ are 256-dimensional vectors whose ith element is the percent of color i. A is a symmetric color similarity matrix with

$$a_{ij} = 1 - \frac{d(c_i, c_j)}{d_{\max}}$$

where c_i and c_j are the ith and jth colors in the color histograms and $d(c_i, c_j)$ is the Mathematical Transform to Munsell (MTM) [24] color distance and $d_{\max}$ is the maximum distance between any two colors. This metric gives the (weighted) length of the difference vector between p and q, weighted by A, which accounts for the perceptual distance between different pairs of colors.

In this paper, we assume that the domain space is the d-dimensional Euclidean space where the dissimilarity is measured by the Euclidean distance weighted by the matrix A. We also assume that QBIC is deployed on "somewhat dynamic" image databases (i.e., where the set of images changes by a small percentage each day). This condition means that the update cost, as well as the retrieval cost, is an issue in determining the usefulness of the system.

Table I gives the summary of symbols and their definitions used in the paper.

D. Structure of the Paper

In Section II, we describe the related work. Section III introduces the LPC-file, with its structure, lower and upper bounds on the distance between a query point $\boldsymbol{q}$ and a vector $\boldsymbol{p}$, the analytic estimation of the vector approximation and k-NN search algorithm. Section IV presents experimental results to assess the performance of the LPC-file. In Section V, we provide a demonstration that shows the effectiveness of our method. In Section VI, we conclude with a discussion of the significance of the work and directions for further research.

TABLE I
SUMMARY OF SYMBOLS AND DEFINITIONS

Symbols	Definitions
d	number of dimensions
N	number of data points in a database
k	number of nearest neighbors to find
p	a database point
q	a query point
a	approximation of p
$k\text{-}NN^{dist}(q)$	distance between the q and the k-th NN q
$k\text{-}NN^{sphere}(q)$	sphere with center q and radius $k\text{-}NN^{dist}(q)$

II. Related Work

Various approaches have been tried for the k-NN search in database systems. Intuitively, it is natural to use MIMs [3], [5], [6], [15], [19] to speed up the k-NN search because objects are often represented by points in a multidimensional data space. However, most MIMs are defeated by the high dimensionality [4], [17], [31]. Motivated by the disadvantages of the state-of-the-art MIMs for high-dimensional data spaces, more recently, other kind of approaches have been investigated to overcome the dimensionality curse. We classify them into four categories:

1) the dimensionality reduction (DR) approach [18], [22];
2) the approximate nearest neighbor (ANN) approach [1], [17], [21], [32];
3) the multiple space-filling curves approach [23], [29];
4) the filter-based approach [31].

In this section, we discuss these new approaches and the problem of the traditional MIMs.

A. Multidimensional Indexing Methods (MIMs)

MIMs work by partitioning the data space, clustering data according to partitions and using the partitions to prune the search space for queries. Unfortunately, while MIMs generally perform well at low-dimensional data spaces, their performance degrades drastically as the dimensionality increases. It is very unlikely that existing MIMs can prune some search space in high-dimesional data spaces and thus most MIMs cannot outperform the sequential scan.

We can summarize the reason as follows. We assume that d-dimensional data set D lies within the unit hypercube $\Omega = [0,1]^d$ and use the L_2 to determine distances. We also assume that data and query points are uniformly distributed within the data space and dimensions are independent. Under these assumptions, we can compute the expected NN distance for a query. Given this distance, an NN query can be reformulated as a spherical range query, such that both the NN query and the reformulated range query visit exactly the same set of disk blocks. Thus, the average cost of a search can be measured by counting the disk blocks which interesect the sphere. In high-dimensional spaces, some effects lead to performance degeneration when applying the conventional partitioning approach. First of all, the data space becomes sparser as the dimensionality increases. Consider a hypercube range query with length s in all dimensions. The probability that a point lies within that range query is given by s^d. It follows directly from this formula that even very large hypercube range queries are not likely to contain a point. At $d = 256$, a range query with length 0.95 in each dimension only selects 0.0002% of data points. From this we can conclude that we can hardly find data points in Ω and, hence, that the data space is very sparsely populated. In order

to maintain a constant density in the data space, the number of data would have to grow exponentially with the dimensionality. This is clearly not the case for real data sets.

Because of the sparsity of the data space caused by the high dimensionality, the distance $k\text{-}NN^{dist}(q)$ between the query point q and the kth NN of q becomes far larger than the length of each dimension of the data space. Fig. 1 shows the expected NN distance as a function of the dimensionality [31]. Notice that the extected NN distance is much larger than the length of a dimension of the data space. As a result, the sphere $k\text{-}NN^{sphere}(q)$ with center q and radius $k\text{-}NN^{dist}(q)$ may intersect every partition in the data space and thus the k-NN search has to visit every partition without pruning anything. Recently, the Pyramid technique [4] was specially devised to circumvent the dimensionality curse. It partitions the data space such that the resulting partitions are shaped like peels of an onion to avoid the situation in which all partitions are intersected by the query region. However, it could not solve the k-NN problem, though it solved the range search problem in high dimensions. In k-NN search, as shown in Fig. 2, the query sphere $k\text{-}NN^{sphere}(q)$ may intersect every partition since $k\text{-}NN^{dist}(q)$ is still far larger than the length of each dimension. This phenomenon is common to most current MIMs. After all, most MIMs have to read not only the entire index file but also the entire data file itself. Therefore, most MIMs cannot outperform the sequential scan in high-dimensional spaces.

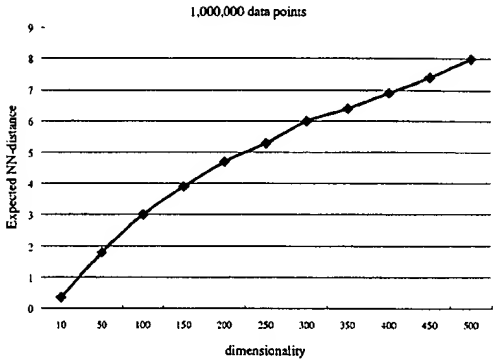


Fig. 1. Expected NN distance as a function of the dimensionality.

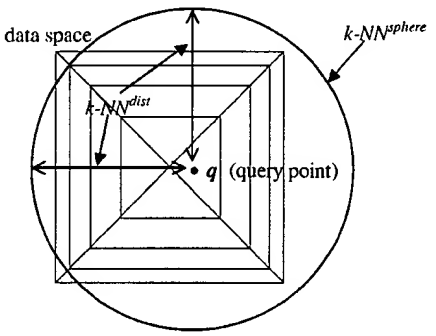


Fig. 2. Space partitioning in the Pyramid technique and the distance $k\text{-}NN^{dist}$ between the query point q and the kth NN of q.

B. Dimensionality Reduction (DR) Approach

The DR approach first condenses most of information in a data set to a few dimensions by applying singular value decomposition (SVD) [30]. The data in the few condensed dimensions are then indexed to support fast retrieval. While the methods based on the DR approach provide a solution to the dimensionality curse, they have several drawbacks.

1) Dimensionality reduction is inevitably accompanied by the loss of precision of query results.
2) They are not readily applicable to dynamic databases because SVD has to be computed *a priori* on the entire data set and the computation is expensive. [18]
3) They work well only when the data is strongly correlated.

Recently, although Kanth *et al.* [18] developed the technique for performing SVD-based DR in dynamic databases, the other limitations of the DR approach still exist.

C. Approximate Nearest Neighbor (ANN) Approach

The idea behind the ANN approach is to retrieve k ANNs faster within a given error bound ε instead of retrieving exact k-NNs. Given a query point q and a distance error $\varepsilon > 0$, a point p is a $(1+\varepsilon)$- ANN of q such that for any other database point p'

$$\|q - p\| \leq (1+\varepsilon)\|q - p'\|.$$

The k-ANN search algorithm works as follows. Let p be the kth NN seen so far and $k\text{-}NN^{dist}(q)$ be its distance to q. As soon as the distance from q to the current cell exceeds $k\text{-}NN^{dist}(q)/(1+\varepsilon)$ (illustrated by the dotted circle in Fig. 2), the search can be terminated and p can be reported as an ANN to q even though the actually nearer point p' has not been searched (see Fig. 3). In this case, p is not the *true* NN.

D. Multiple Space-Filling Curves Approach

This approach is also a kind of approximate approach in the sense that it gives up some accuracy of query results. The multiple space-filling curves approach [23], [29] orders the d-dimensional space in many ways, with a set of space-filling curves, each constituting a mapping from $R^d \rightarrow R^1$. This mapping provides a position along the curve for any d-dimensional vector. In essence, this gives a linear ordering of all points in the data set. Unlike projections, in this linear ordering, close points along the space-filling curve tend to correspond to close points in the d-dimensional data space. Therefore, when a query point is mapped to the space-filling curve, one can perform a range search for nearby points along the curve to find near neighbors in the data space. However, due to the nature of the $R^d \rightarrow R^1$ mapping, some near neighbors of the query point may be mapped far apart along a single curve. To make sure that these points are not overlooked, multiple space-filling curves are used, based on different mappings from $R^d \rightarrow R^1$. The set of candidate NNs is formed from the union of sets of points in small neighborhoods (ranges) from all of the curves.

Like the ANN approach, although this approach improves the performance of the k-NN search, it is possible that some near-neighbors may be omitted during a search. To improve the chances of finding all kNNs, more curves have to be used or a larger neighborhood on each curve has to be scanned. However,

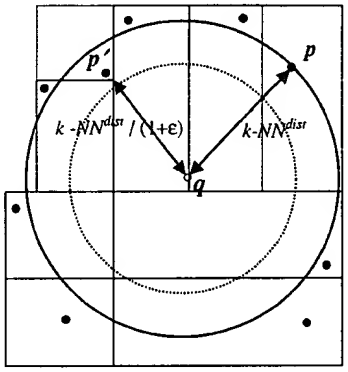


Fig. 3. Overview of ANN search algorithm.

since both of these methods increase the size of the candidate set, they diminish the benefits of the approximate approach.

E. Filter-Based Approach

The filter-based approach overcomes the dimensionality curse by filtering the vectors so that only a small fraction of them must be visited during a search. Among the approaches that aim at overcoming the dimensionality curse, this approach is the only one that outputs the *exact* kNNs and the VA-file [31] is the only one classified into this category.

The VA-file divides the data space into 2^b rectangular cells where b denotes a user specified number of bits. Instead of hierarchically organizing these cells like in MIMs, the VA-file allocates a unique bit-string of length b to each cell and approximates data points that fall into a cell by that bit-string. The VA-file itself is simply an array of these compact approximations of data points. The k-NN queries are processed by first scanning the entire approximation file and by excluding (filtering) the vast majority of vectors from the search based only on these approximations.

The performance of the VA-file largely depends on the filtering power of the method and, in turn, the filtering power relies on the preciseness of the approximation. Although the VA-file provided a solution to the dimensionality curse and outperformed the sequential scan and most MIMs in high dimensions, it has some drawbacks. The major drawbacks of the VA-file are: more bits are needed for the approximations in proportion to the dimensionality to enhance the filtering power; and the filtering power decreases severely for clustered data, such as image data.

The filtering approach benefits from the *sparseness* of the high-dimensional data space as opposed to the partitioning or clustering methods (i.e., the traditional MIMs). By *clustering* in this paper, we mean that several vectors (data points) lie in the same cell. In fact, it is very unlikely that several vectors lie in the same cell, as described in Section II-A. Furthermore, the vast majority of the cells must be empty since the number of cells 2^b is much larger than the number of vectors in a database.

If several vectors use the same approximation, the filtering power of the approximation decreases since we cannot discriminate among the vectors that share the approximation. Let us consider two extreme cases:

1) using the roughest approximation such that all vectors lie in one cell;
2) using the finest approximation (this is the case that vectors themselves are used without being approximated).

In the case of 1), we cannot filter out any vectors during the search because every approximation is the same for vectors. In contrast, in the case of 2), vectors not to be included in the answer set are completely filtered out because every approximation (in fact, vector itself) is completely discriminated. That is, the case of 2) is the same as the sequential scan. If we use rougher approximations, more vectors lie in the same cell and thus the filtering power of approximation decreases. The performance of the filtering approach depends on the filtering power of the method and, in turn, the filtering power relies on the preciseness of the approximation. If several vectors lie in the same cell, a finer approximation must be used to improve the filtering power of the approximation. However, as we use a finer approximation, the filtering method approaches to the sequential scan.

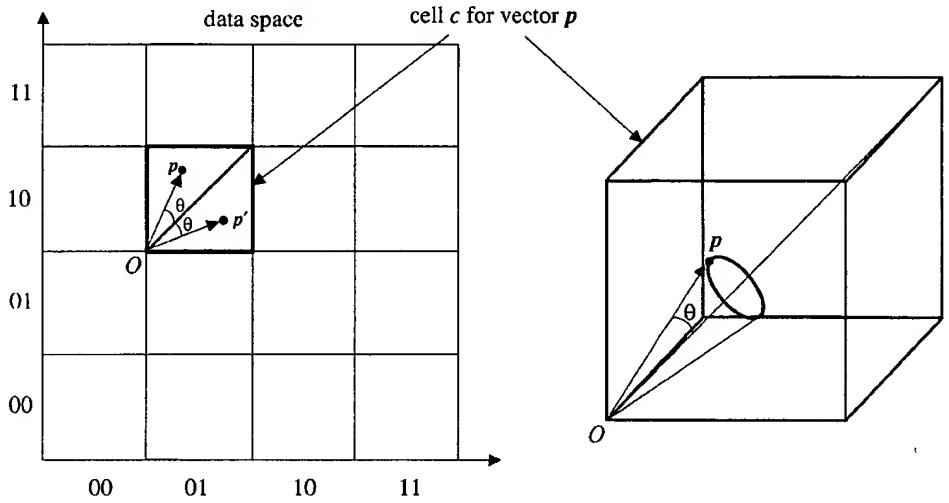
In image databases, images tend to cluster in certain cells. As a result of this clustering pattern, for the VA-file, it becomes difficult to filter out noncandidate approximations in clustered image data sets. To enhance the filtering rate, the VA-file must use more bits for approximations. However, the performace of the VA-file converges to that of the sequential scan or degenerates to being worse than that as the number of bits used for approximations increases. To overcome these drawbacks, we propose the LPC-file that increases the filtering rate of the filter-based approach by only adding a small amount of additional information. The amount of additional information is independent of the dimensionality. Therefore, it does not increase with the dimensionality and the performance of the LPC-file does not deteriorates below that of the sequential scan.

III. THE LPC-FILE

The LPC-file takes a filter-based approach in common with the VA-file. Thus, first, the vector space is partitioned into rectangular cells and these cells are used to generate bit-encoded approximations for each vector. The method used for this stage is similar to that of the VA-file. However, unlike the VA-file which simply uses more bits to each dimension to increase the discriminatory power of the approximation, the LPC-file enhances it by adding polar coordinate information of the vector to the approximation.

A. Structure of the LPC-File

The LPC-file is a flat file organization of geometric approximations to data points. The design goal of the LPC-file is to maximize the filtering power of the filter-based method by using minimal information. If we simply increase the number of bits to represent the cell approximation as the VA-file does, the amount of data to be read from the disk also increases with the dimensionality even though the filtering power increases. There is thus a tradeoff between the filtering power and the number of disk accesses in query processing time. We compromise this tradeoff by adding polar coordinates of vectors to the cell approximation. The polar coordinate information is independent of the dimensionality and thus we need not use more bits to represent it as



(a) 2-dimensional space (b) 3-dimensional space

Fig. 4. Vector p and its approximation in the LPC-file.

the dimensionality increases. It is sufficient to use only 3 B for polar coordinates (2 B for radius and 1 B for angle). It is desirable to use dimensionality-independent information if possible because we deal with high-dimensional vectors, e.g., over 100 dimensions. For $N = 1{,}000{,}000$ and $d = 256$, if we use one more bit per dimension for the approximation, the amount of data that has to be read from the disk in query processing time increases by 32 MB. If we incorporate the polar coordinate information into the cell approximation instead of using one more bit per dimension, we need to read only additional 3 MB data from the disk. The saving of disk access we achieve is a factor of 10.

For each vector $\boldsymbol{p}_i$, $i \in \{1, \ldots, N\}$, an approximation $\boldsymbol{a}_i$ is derived, An approximation is generated as follows. The first step is to assign the same number of bits b to each dimension and to divide the whole data space into 2^{bd} cells. Thus the length of each side of a cell is same. Typically b is a small integer between "4" and "8" depending on the dimensionality and the data distribution. The cell is simply represented by the concatenation of the binary bit patterns for each dimension in turn. Fig. 4(a) shows an example in two dimensions: the cell c is represented by the sequence of bits (01 10) where $d = 2$ and $b = 2$.

The second step is to represent the vector $\boldsymbol{p}$ using the polar coordinates (r, θ) within the cell in which $\boldsymbol{p}$ lies. As illustrated in Fig. 4(a), the local origin O of each cell is determined by the lower left corner of the cell. The radius r is computed by the distance between the local origin O and the vector $\boldsymbol{p}$. The angle θ is computed by the angle between the vector $\boldsymbol{p}$ and the diagonal from the local origin to the opposite corner. As a result of this approximation, the vector $\boldsymbol{p}$ is represented by the triplet $\boldsymbol{a} = \langle c, r, \theta \rangle$, where c, r, and θ denote the approximation cell, the radius, and the angle of $\boldsymbol{p}$, respectively. The complete LPC-file is an array of approximations for all vectors.

An approximation of the LPC-file represents a set of points which have radius r and angle θ within the cell c. In a two-dimensional space, there are two points $\boldsymbol{p}$ and $\boldsymbol{p}'$ which have the same polar coordinates (r, θ) and are symmetric with respect to the diagonal [as illustrated in Fig. 4(a)]. On the other hand, the VA-file represents the approximation by the whole cell c. In a three-dimensional (3-D) space, the approximation of the LPC-file is represented by a circle around the diagonal as shown in Fig. 4(b), while the approximation of the VA-file is the cube itself. In higher dimensions, the approximation of the LPC-file is a set of points on a hypersphere.

B. Lower and Upper Bounds on Distance

Based on approximations, we now derive the bounds on the distance between a query point and a vector to restrict the search space during the k-NN search. For a query vector $\boldsymbol{q}$, an object vector $\boldsymbol{p}$ and a distance function L_2, the approximation $\boldsymbol{a}$ determines a lower bound $d_{\min}$ and an upper bound $d_{\max}$ such that

$$d_{\min} \leq L_2(\boldsymbol{p}, \boldsymbol{q}) \leq d_{\max}.$$

This is sketched in Fig. 5. To obtain the distance $L_2(\boldsymbol{p}, \boldsymbol{q})$ between the vector $\boldsymbol{q}$ and the vector $\boldsymbol{p}$, we close the triangle OAB and apply the cosine rule. Thus $\mathrm{AB}^2 = \mathrm{OA}^2 + \mathrm{OB}^2 - 2\mathrm{OA} \cdot \mathrm{OB} \cos\phi$ or

$$|\boldsymbol{p} - \boldsymbol{q}|^2 = |\boldsymbol{p}|^2 + |\boldsymbol{q}|^2 - 2|\boldsymbol{p}||\boldsymbol{q}| \cos\phi \tag{1}$$

where ϕ is the angle $\angle$AOB. Expressing these moduli in terms of the Cartesian components, we obtain the equation

$$\sum_{i=1}^{d} (p_i - q_i)^2 = \sum_{i=1}^{d} p_i^2 + \sum_{i=1}^{d} q_i^2 - 2|\boldsymbol{p}||\boldsymbol{q} \cos\phi.$$

The $\cos\phi$ between two vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ then is

$$\cos\phi = \frac{\sum_{i=1}^{d} p_i \cdot q_i}{|\boldsymbol{p}||\boldsymbol{q}|}$$

where d is the number of dimensions.

From (1), the lower bound $d_{\min}$ and upper bound $d_{\max}$ are determined when the value of the $\cos\phi$ is maximum and minimum, respectively. In other words, $d_{\min}$ and $d_{\max}$ are deter-

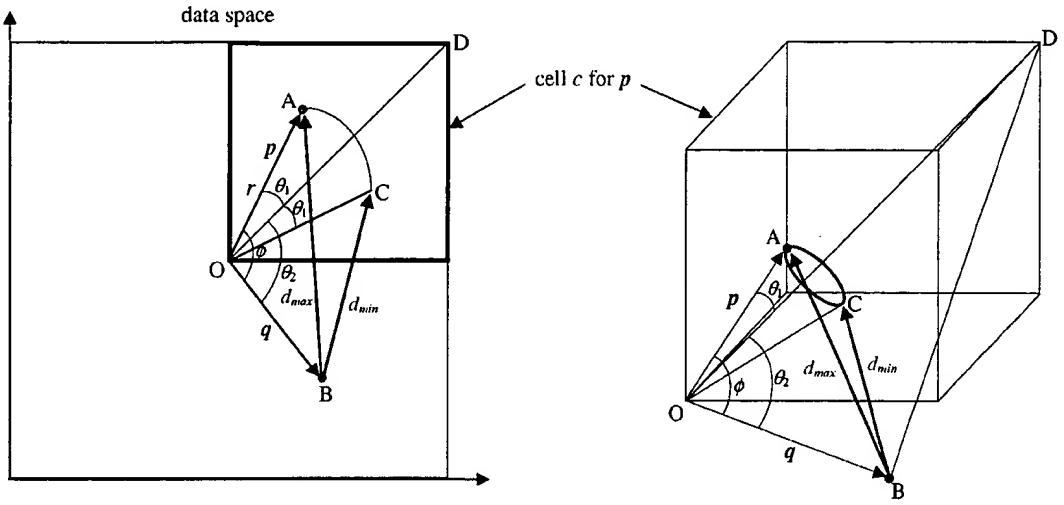


Fig. 5. Lower bound $d_{\min}$ and upper bound $d_{\max}$ for $L_2(\boldsymbol{p}, \boldsymbol{q})$ guaranteeing $d_{\min} \leq L_2(\boldsymbol{p}, \boldsymbol{q}) \leq d_{\max}$.

mined when the angle $\phi(0° \leq \phi \leq 180°)$ between two vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ is minimum and maximum, respectively. The minimum angle and the maximum angle between two vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ are determined by $|\theta_1 - \theta_2|$ and $(\theta_1 + \theta_2)$, respectively, where $\theta_1 (= \angle \mathrm{AOD})$ is the angle between the vector $\boldsymbol{p}$ and the diagonal of the cell in which $\boldsymbol{p}$ lies and $\theta_2 (= \angle \mathrm{BOD})$ is the angle between the vector $\boldsymbol{q}$ and the diagonal of the cell in which $\boldsymbol{p}$ lies (as illustrated in Fig. 5). Based on this information, the lower and upper bounds $d_{\min}$ and $d_{\max}$ are determined by the equations

$$d_{\min} = |\boldsymbol{p}|^2 + |\boldsymbol{q}|^2 - 2|\boldsymbol{p}||\boldsymbol{q}| \cos|\theta_1 - \theta_2|$$
$$d_{\max} = |\boldsymbol{p}|^2 + |\boldsymbol{q}|^2 - 2|\boldsymbol{p}||\boldsymbol{q}| \cos(\theta_1 - \theta_2).$$

Without loss of generality, these properties hold in any dimension. In a 3-D case, for example, the approximation of $\boldsymbol{p}$ is the circle around the diagonal OD as shown in Fig. 5(b). Thus, A (the point having $d_{\max}$) and C (the point having $d_{\min}$) are the points at which the circle intersects with the OBD plane. In other words, points O, A, B, C, and D lie on the same plane.

Fig. 6 compares the lower and upper bounds of the LPC-file and the VA-file. The lower and upper bounds of the VA-file are simply the shortest and longest distance from a query point to the cell. To the contrary, the lower bound of the LPC-file is larger than that of the VA-file and the upper bound of the LPC-file is smaller than that of the VA-file. In other words, The difference diff $= d_{\max} - d_{\min}$ of the LPC-file is smaller than that of the VA-file. It shows the tightness of the approximation of the LPC-file.

C. Estimation of the Approximation of the LPC-File

Given a query $\boldsymbol{q}$ and the approximations of the LPC-file and the VA-file, we can approximately estimate the *precision* of the approximations of both methods by the size of the approximations. The precision of the approximation measures how much close the approximation is to the vector. Smaller size of the approximation represents higher precision of the approximation. Smaller approximations usually make the difference diff $= d_{\max} - d_{\min}$ smaller. If we assume that the data is uniformly distributed and the length s of every dimension of the vector approximation cell is the same, then the size of the approximation cell of the VA-file is s^d. The size of the approximation of the LPC-file in a d-dimensional space is defined by the derivative $\mathrm{Vol}'(x)$ of the volume $\mathrm{Vol}(x)$ of the $(d-1)$-dimensional hypersphere. The volume of the d-dimensional hypersphere with radius x is given by

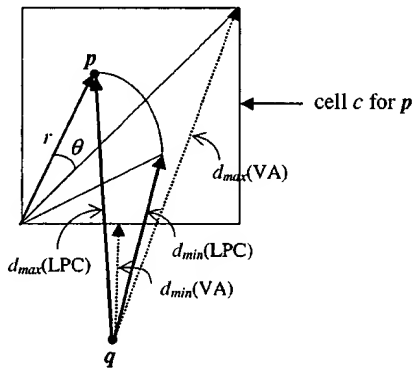


Fig. 6. Lower bound $d_{\min}$(LPC) and upper bound $d_{\max}$(LPC) for the LPC-file and lower bound $d_{\min}$(VA) and upper bound $d_{\max}$(VA) for the VA-file.

$$\mathrm{Vol}(Sp^d(x)) = \frac{\sqrt{\pi^d}}{\Gamma(\frac{d}{2}+1)} \cdot x^d$$
$$\text{with } \Gamma(x+1) = x \cdot \Gamma(x),\ \Gamma(1) = 1$$
$$\text{and } \Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}.$$

Then the size of the approximation of the LPC-file is given as follows when we assume the radius of the hypersphere is r:

$$Vol'(Sp^{d-1}(r)) = \frac{\sqrt{\pi^{d-1}}}{\Gamma(\frac{d-1}{2}+1)} \cdot (d-1) r^{d-2} = \frac{2^{d-1}\pi^{(d-1)/2}}{(d-1)!} \cdot (d-1) r^{d-2}. \quad (2)$$

To compare the size of the LPC-file approximation with the size s^d of the VA-file vector approximation, we performed sev-

eral experiments. From our experimental results using 256-dimensional uniform data set, the average length of radius r of the $(d-1)$-dimensional hypersphere was 0.58 when we normalize the length of side s of the approximation cell to 1.0. When we set $s = 1.0$, $r = 0.58$ and $d = 256$ in the hypervolume s^d of the VA-file approximation and (2), we observed that the hypervolume of the approximation of the LPC-file is far smaller than that of the VA-file. Moreover, our experimental results given in Section IV-A show that the $d_{\min}$ of the LPC-file is always larger than that of the VA-file and the $d_{\max}$ of the LPC-file is always smaller than that of the VA-file. In other words, it demonstrates the tightness of the LPC-file approximation.

D. Search Algorithm

The search algorithm consists of two stages. In the first stage, it collects vectors to make a candidate set. To do that it scans approximations and the lower and upper bounds $d_{\min}$ and $d_{\max}$ are computed for each vector. If the approximation is found whose $d_{\min}$ exceeds the distance k-NN$^{dist}(\boldsymbol{q})$ of the kth NN encountered so far, then the corresponding vector can be eliminated since k better candidates have already been found. In our experiments, on the average, more than 99% of the vectors were eliminated during this first filtering stage. In the second stage, it refines the candidate set by visiting real vectors themselves in increasing order of $d_{\min}$. In this stage, not all remaining candidates are visited. Rather, this stage ends when an approximation is encountered whose lower bound exceeds or equals the kth distance k-NN$^{dist}(\boldsymbol{q})$ in the answer set and the final k nearest vectors in the answer set are the search result. The experimental results showed that, on the average, more than 99.9% of the vectors were eliminated through this second refinement stage.

The algorithm **k_NN_Search** is described in C-like pseudocode in Fig. 7. The array *knn* is used to maintain the nearest k vectors encountered so far and their distances to the query vector $\boldsymbol{q}$. These are maintained in the order of increasing distance. The priority queue *cand_list* is used to maintain the candidate set and the data structure supporting this procedure is *min heap*[16].

An important performance factor in this two-stage algorithm is that the disk access pattern in the stage 1 is sequential, whereas that is random in the stage 2. Thus disk accesses occurred in the stage 2 have greater effects on the search performance than those in the stage 1.

```
Algorithm k_NN_Search (q: vector, k: integer)
{
// Variables used in the algorithm
// k-NN^dist(q): the k-th largest distance between the query
//              vector q and the vectors p encountered so far
// N: the number of vectors in the database
// knn: answer list to maintain the nearest k vectors encoun-
// tered so far and their distances to the query vector q
// cand_list: min heap to maintain the candidate set
// c: a candidate to insert into the cand_list
// c.oid: identifier assigned to uniquely identify the candidate c
// nn: a near neighbor to insert into the knn
// MAX: a value that exceeds the possible largest distance
//       between any two points within the database

  // Stage 1
  for i:=0 to k do {
      knn[i].dist := MAX;
  }
  k-NN^dist(q) := MAX;
  For every approximation a in the approximation set {
    Compute the lower and upper bounds a.d_min and a.d_max
      of a.
    if (a.d_min ≤ k-NN^dist(q)) {
      Insert [c := {a.oid, a.d_min, a.d_max}] to the candidate
        set cand_list;
      if (c.d_max < k-NN^dist(q)) {
        // The following is an ordered insertion in the knn
        // array, i.e., the new element is inserted into the
        // correct position with respect to the distance in knn.
          Insert the near neighbor [nn := {oid = c.oid, dist =
            c.d_max}] to the answer set knn;
        // Update k-NN^dist(q) after each insertion, if it gets
        // smaller.
          k-NN^dist(q) := the distance of the k-th nearest
            neighbor in the answer set knn;
      }
    }
  }

  // Stage 2
  for i:=0 to k do {
    knn[i].dist := MAX;
  }
  while (get the candidate c from the candidate set and
          c.d_min ≤ k-NN^dist(q)) do {
    Read vector p corresponding to the c.oid;
    if (L2(p, q) < k-NN^dist(q)) {
      Insert the near neighbor [nn := {oid = c.oid, dist =
        L2(p, q)}] to the answer set knn.
      k-NN^dist(q) := the distance of the k-th nearest neighbor
        in the answer set knn;
    }
  }
}
```

Fig. 7. k-NN search algorithm of the LPC-file.

IV. Performance Evaluation

To demonstrate the practical effectiveness of the LPC-file, we performed an extensive experimental evaluation of the LPC-file and compared it to the competitors: the VA-file and the sequential scan. Our experiments have been computed under the Microsoft Windows NT 4.0 on Intel Pentium II 266 MHz Processor with 192 MB of main memory. For our tests we used two groups of synthetic data sets:

- *random data set* which follows the random distribution;
- *skewed data set* which follows the skewed distribution according to Zipf's law [20].

The Zipf distribution is defined as follows and the value of z we used is 0.7:

$$f(i) = \frac{\frac{1}{i^z}}{\sum_{j=1}^{n} \frac{1}{j^z}}, \quad i = 1, 2, \ldots, n.$$

We also used *real* data set from the QBIC image database.

	d_{min}(VA)	d_{min}(LPC)	d_{max}(LPC)	d_{max}(VA)	*diff*(LPC)	*diff*(VA)
random(6)	6.378	6.407	6.554	6.582	1.0	1.39
skewed(6)	7.463	7.512	7.642	7.670	1.0	1.59
real (8)	4.702	4.710	4.745	4.755	1.0	1.50

(a)

	d_{min}(VA)	d_{min}(LPC)	d_{max}(LPC)	d_{max}(VA)	*diff*(LPC)	*diff*(VA)
random(6)	4.557	4.577	4.681	4.701	1.0	1.40
skewed(6)	5.380	5.415	5.506	5.526	1.0	1.61
real (8)	3.313	3.319	3.344	3.351	1.0	1.50

(b)

	d_{min}(VA)	d_{min}(LPC)	d_{max}(LPC)	d_{max}(VA)	*diff*(LPC)	*diff*(VA)
random(6)	3.203	3.218	3.290	3.305	1.0	1.41
skewed(6)	3.771	3.796	3.860	3.874	1.0	1.62
real (8)	2.315	2.319	2.336	2.341	1.0	1.49

(c)

	d_{min}(VA)	d_{min}(LPC)	d_{max}(LPC)	d_{max}(VA)	*diff*(LPC)	*diff*(VA)
random(4)	2.128	2.171	2.371	2.417	1.0	1.45
skewed(4)	2.499	2.580	2.756	2.792	1.0	1.66
real (8)	1.617	1.620	1.632	1.636	1.0	1.53

(d)

	d_{min}(VA)	d_{min}(LPC)	d_{max}(LPC)	d_{max}(VA)	*diff*(LPC)	*diff*(VA)
random(4)	1.475	1.509	1.644	1.680	1.0	1.52
skewed(4)	1.699	1.757	1.880	1.907	1.0	1.70
real (8)	1.114	1.116	1.124	1.127	1.0	1.57

(e)

Fig. 8. Lower and upper bounds and their differences in (a) 256 dimensions, (b) 128 dimensions, (c) 64 dimensions, (d) 32 dimensions, and (e) 16 dimensions.

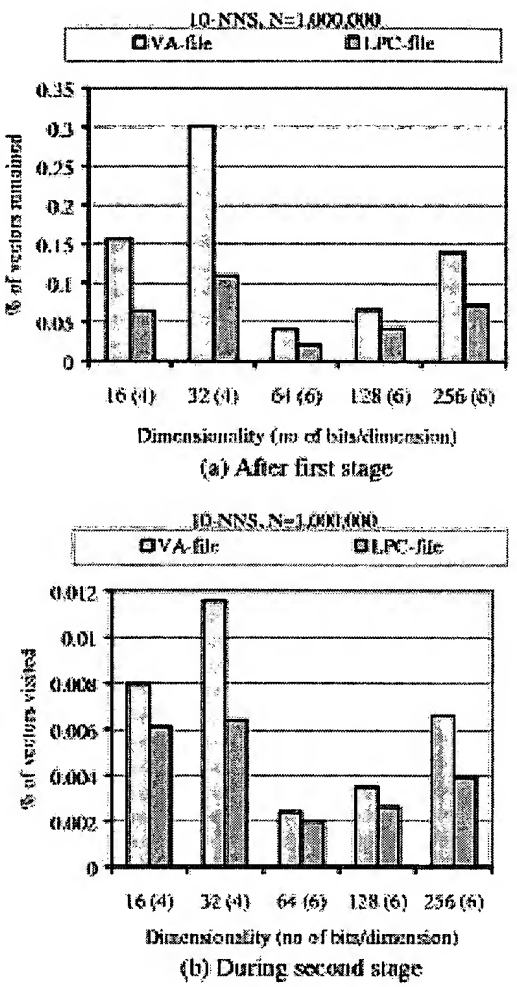


Fig. 9. Vector selectivity on random data set.

- 13 724 256-color images of U.S. stamps and photos: Stamps often come in series (e.g., states, birds, flowers) with common colors and related designs and the U.S. Post Office has often used similar colors for many long-running stamps. As a result, this real image data set shows *clustered distribution*. Moreover, before storing images into the database, each image $\boldsymbol{p}$ is transformed to $\boldsymbol{p}^T A \boldsymbol{p}$, where $\boldsymbol{p}$ represents 256-dimensional color histogram and A is a color similarity matrix described in Section I-C.

For a number of experiments we performed, a data set containing far more than 13 724 vectors was required. To obtain this larger database, the 13 724-vector data set was synthetically scaled up to one million, while retaining the original distribution of the real image data set within each dimension. To generate a new image vector v, for each dimension i, $0 \leq i \leq 255$, of v, we randomly selected two out of 13 724 original vectors and averaged their values in dimension i and used it as the value for the dimension i of v.

In all experiments, the Euclidean distance metric L_2 was used and the number of NNs to return was always ten, i.e., $k = 10$. The page size used in the experiment was 4 KB. Each data set has 1 000 000 data points in dimensions 16, 32, 64, 128, and 256 and 100 random 10-NN queries were processed and the results were averaged. The number of bytes of the data points used in each dimension was 2. The number of bits per dimension of the approximation cell used in the VA-file and LPC-file were 4, 6, and 8, depending on the dimensionality and data distributions. Thus the compression ratio of approximations of the VA-file and the LPC-file used in the experiments is on the order of $\frac{1}{4}$ to $\frac{1}{2}$ over the real vectors. The indexing techniques based on the filtering approach try to achieve the performance improvement over the sequential scan to the extent of the compression ratio.

A. Distance Bounds Experiments

The performance of the filtering method depends largely on the distance bounds $d_{\min}$ and $d_{\max}$ between a query point and a vector. The difference diff $= d_{\max} - d_{\min}$ can be used as an estimator of the precision of an approximation of the filtering method. Smaller difference means that the approximation is more precise or tighter.

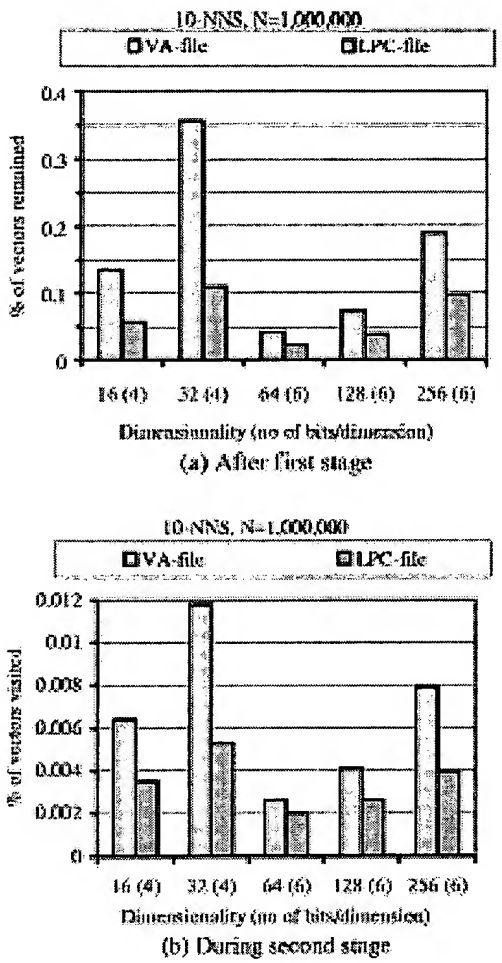


Fig. 10. Vector selectivity on skewed (Zipf) data set.

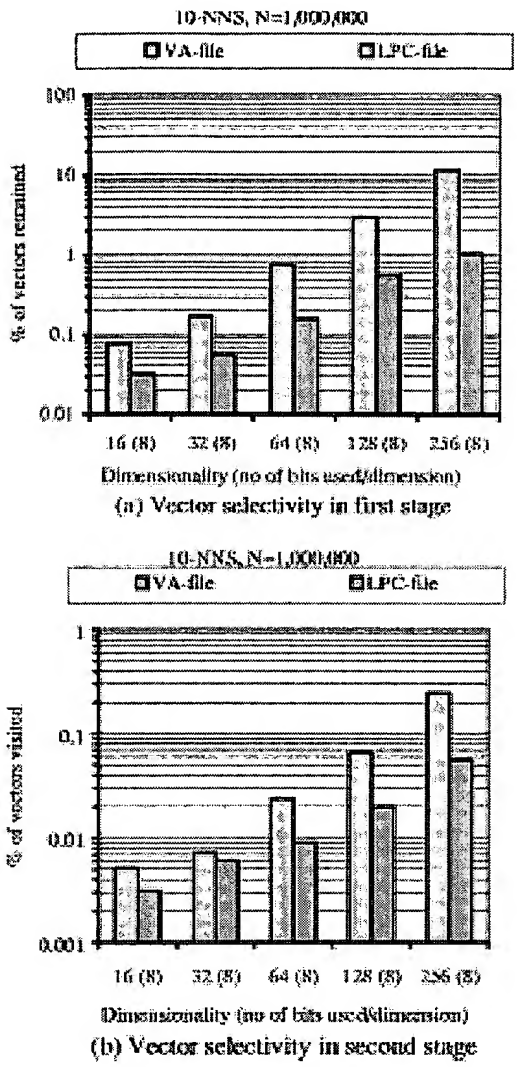


Fig. 11. Vector selectivity in real image data.

Fig. 8 shows the average $d_{\min}$ and $d_{\max}$ values and their differences for the VA-file and LPC-file approximations computed from 100 random query points given in the dimensionalities of 256, 128, 64, 32, and 16. The values shown were computed by normalizing the length of a dimension to 1.0. As we mentioned in Section II-A, the values $d_{\min}$ and $d_{\max}$ are far larger than the length of a dimension in high-dimensional spaces. The first column of each table denotes the data distribution and the number of bits per dimension for approximation cell used in the experiments.

To simplify the comparison of the difference value diff for the LPC-file and the VA-file, we normalized the diff value for the LPC-file to 1.0, which was represented by diff(LPC) in each table and the relative diff value of the bounds of the VA-file were reported, which was represented by diff(VA) in the table. As we can see in the results, the following condition is always satisfied:

$$d_{\min}(\text{VA-file}) \leq d_{\min}(\text{LPC-file})$$

and

$$d_{\max}(\text{LPC-file}) \leq d_{\max}(\text{VA-file}). \quad (3)$$

Equation (3) is the basis of the superiority of the LPC-file over the VA-file. Also, in every dimension, the diff(LPC) value of the LPC-file is uniformly 40–70% less than the value diff(VA) of the VA-file.

B. Selectivity Experiments

The *vector selectivity* is defined as the ratio of the number of vectors visited to the total number of vectors in a database [31]. The vector selectivity is a good performance estimator because the performance of the k-NN search depends largely on the number of disk blocks visited. Moreover, the number of vectors sharing the same disk block decreases as the dimensionality increases and the disk access pattern of the k-NN search algorithm is random. Thus, in fact, the vector selectivity can be an estimator for the number of random disk accesses for the k-NN search. Figs. 9–11 show the vector selectivity when the random, skewed and real data sets were used, respectively. The vector selectivity was measured as a function of the dimensionality and the number of bits used for the cell. In each figure, the horizontal

axis shows the dimensionality of data set and the number of bits, which is represented within the parenthesis, used to represent each dimension of the cell. The vertical axis denotes the vector selectivity. In the case of the LPC-file in the synthetic (random and skewed) data set, less than 0.1% of vectors remained after the first stage of the **k_NN_Search** algorithm and less than 0.01% of vectors were actually visited in the second stage. The vector selectivity of the LPC-file in each stage is about a half of that of the VA-file and the ratio of the difference between their selectivities increases as the dimensionality increases. This selectivity is directly reflected in the cost of disk access. In fact, in a database with $N = 1{,}000{,}000$ vectors, the LPC-file visits less than 50 vectors on the average. In Figs. 9 and 10, the selectivities in dimensions 16 and 32 are higher than those in higher dimensions because, in those lower dimensions, less bits were used in the cell approximation. In lower dimensions, it is sufficient to use less bits to maintain the precision of the cell approximation. Fig. 11 shows the vector selectivity measured in a real image database. As mentioned before, this image set is so strongly clustered. In this case, the filtering power of the approximated cell of the VA-file decreases drastically unless even more bits are used for the cell. In fact, this is almost the worst case of distributions because the probability that vectors share the same cell is high and thus the discriminatory power of the approximation decreases severely. To the contrary, the LPC-file demonstrates its robustness even in this bad kind of distributions, as shown in Fig. 11. Note that the scale of the y-axis of Fig. 11 is logarithmic. The vector selectivity of the LPC-file is far less than that of the VA-file.

C. Elapsed Time Experiments

While the selectivity experimental results above are encouraging, they do not fully reflect the reality. To demonstrate the practical effectiveness of the LPC-file, we performed a number of timing experiments. Fig. 12 shows the elapsed time of the k-NN search. The Scan algorithm is a simple sequential scan of the vectors themselves, maintaining a ranked list of the k-NN vectors encountered so far. The important advantages of the sequential scan are:

1) it does not need to read index file;
2) it is very simple;
3) it can perform sequential disk accesses rather than random disk accesses.

While the I/O patterns generated by the k-NN search algorithm are inherently random, the sequential scan can save much disk start-up time to begin the read. Because of these factors, a well-tuned sequential scan frequently outperforms more sophisticated methods which frequently generate the random disk access. Moreover, most MIMs could not outperform the sequential scan in high dimensions. Therefore, instead of MIMs, the sequential scan can be used as the yardstick for performance comparison in high dimensions.

Fig. 12 shows the elapsed time of the k-NN search, where $k = 10$, for the Scan, the VA-file, and the LPC-file. In the synthetic (random and skewed) data set, the LPC-file outperforms the VA-file and the Scan by a factor of 2 and 3 on the average, respectively. In real data set, the performance of the VA-file decreases drastically as soon as its vector selectivity falls below a certain threshold. In a 256-dimensional real image set, the performance of the VA-file using 8 bits per dimension for a cell degenerates close to that of the Scan. Even in this worst case of data distributions, the LPC-file shows a performance superiority to the Scan, reflecting the reduction of data performed by the approximation.

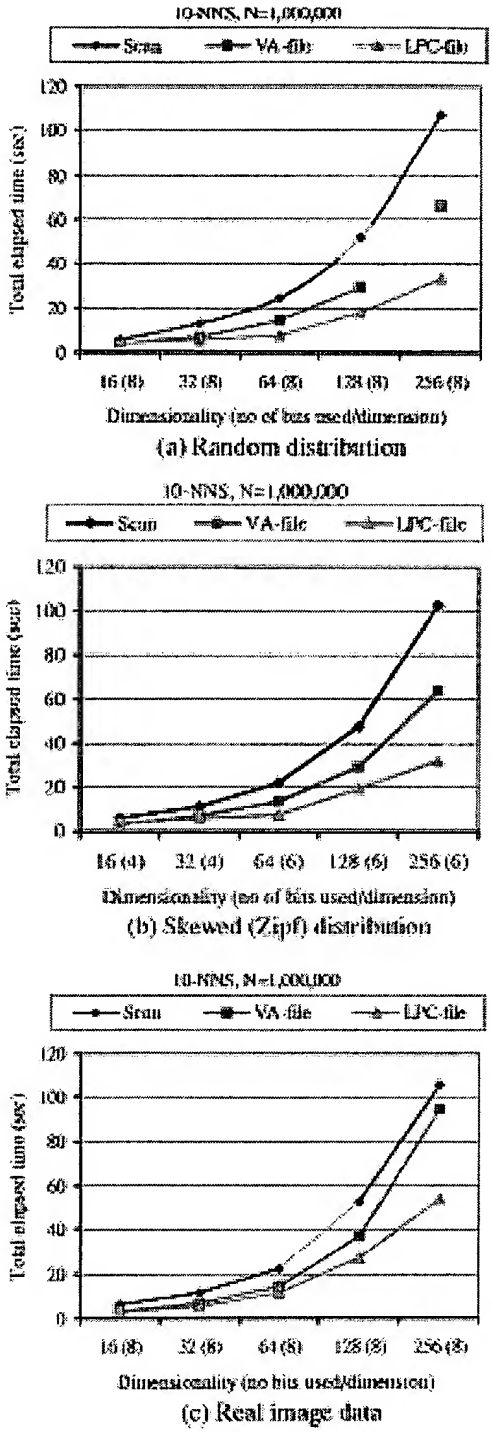


Fig. 12. Total elapsed time.

Fig. 13. Results of query on U.S. stamp images. The query was to find images similar to the one on the upper left.

Comparing the experimental results on the real data set with those on the synthetic data sets, both of the LPC-file and the VA-file have worse performance on the real data set than on the synthetic data sets. Because the real data set is strongly clustered on the data space, it is more likely that vectors have same approximations. This degenerates the performance of both of the LPC-file and the VA-file. However, the performance degeneration of the LPC-file is much lower than that of the VA-file because of the tightness of approximation of the LPC-file.

D. Demos Using U.S. Stamps

A sample application using images of U.S. stamps demonstrates the power of the LPC-file on human-generated commercial quality artwork. Content-based queries on the stamp image database often give excellent results. A typical query (for images matching the stamp in the upper left) is shown in Fig. 13. The original stamps images are from a CR-ROM from Walnut Creek CD-ROM.

V. Conclusions

In this paper, we proposed a new indexing technique, the LPC-file, for NN searches in high-dimensional image databases. It is based on approximating vectors by the polar coordinates on the local cell. It improves the k-NN search performance significantly in high dimensions by the discriminatory power of the polar coordinate information. Experimental results based on the distance bound, the vector selectivity and the elapsed time are very promising. In high dimensions, the LPC-file outperforms the competitors such as the VA-file and the sequential scan with a factor of 2 or 3 in total elapsed time to process k-NN queries.

In fact, it has been urgent to develop a technique to speed up the similarity search in high dimensions. Most of newly emerging applications such as multimedia, world-wide web, document retrieval and data mining deal with nontraditional data represented in vectors in high dimensions. However, most methods have been defeated by the high dimensionality. Although some methods have been proposed to improve the performance, they gave up the query precision.

In short, the LPC-file is an efficient indexing technique that outputs *exact* results for the k-NN search in a high-dimensional data space. Our future activity will concentrate on improving the performance of the LPC-file by incorporating the advantages of other NN approaches, such as the DR approach and the ANN approach.

References

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," *J. ACM*, vol. 45, no. 6, pp. 891–923, 1998.

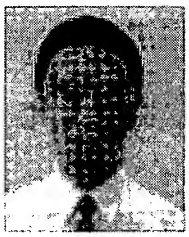
[2] M. Atkinson, J. Sack, N. Santoro, and T. Strothotte, "Min-max heaps and generalized priority queues," *Commun. ACM*, vol. 29, no. 10, pp. 996–1000, 1986.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R^*-tree: An efficient and robust access method for points and rectangles," in *Proc.ACM SIGMOD Int. Conf. Management of Data*, 1990, pp. 322–331.

[4] S. Berchtold, C. Bochm, and H.-P. Kriegel, "The pyramid-technique: Toward breaking the curse of dimensionality," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 1998, pp. 142–153.
[5] S. Berchtold, D. A. Keim, and H.-P. Kriegel, "The X-tree: An index structure for high-dimensional data," in *Proc. 22nd Int. Confe. Very Large Data Bases*, 1996, pp. 28–39.
[6] G.-H. Cha and C.-W. Chung, "A new indexing scheme for content-based image retrieval," *Multimedia Tools Applicat.*, vol. 6, no. 3, pp. 263–288, May 1998.
[7] D. Comer, "The ubiquitous B-tree," *ACM Comput. Surv.*, vol. 11, no. 2, pp. 121–137, 1979.
[8] S. Cost and S. Salzberg, "A weighted nearest neighbor algorithm for learning with symbolic features," in *Mach. Learn.*, 1993, vol. 10, pp. 57–67.
[9] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 21–27, 1967.
[10] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *J. Amer. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.
[11] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
[12] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*. Cambridge, MA: AAAI Press/MIT Press, 1996.
[13] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The QBIC system," *IEEE Comput.*, vol. 28, pp. 23–32, 1995.
[14] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification," in *Proc. Int. Conf. Knowledge Discovery and Data Mining*, 1995, pp. 142–149.
[15] A. Henrich, "The LSD^h-tree: An access structure for feature vectors," in *Proc. 14th Int. Conf. Data Engineering*, 1998, pp. 362–369.
[16] E. Horowitz, S. Sahni, and D. Mehta, *Fundamentals of Data Structures in C++*. Rockville, MD: Computer Science, 1995.
[17] P. Indyk and R. Motwani, "Approximate nearest neighbors: Toward removing the curse of dimensionality," in *Proc. ACM Symp. Theory of Computing*, 1998, pp. 604–613.
[18] K. V. R. Kanth, D. Agrawal, and A. Singh, "Dimensionality reduction for similarity searching in dynamic databases," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 1998, pp. 166–176.
[19] N. Katayama and S. Satoh, "The SR-tree: An index structure for high-dimensional nearest neighbor queries," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 1997, pp. 369–380.
[20] D. Knuth, *The Art of Computer Programming*. Reading, MA: Addison-Wesley, 1973, vol. 3, Sorting and Searching.
[21] E. Kushilevitz, R. Ostrovsky, and Y. Rabani, "Efficient search for approximate nearest neighbor in high-dimensional spaces," in *Proc. ACM Symp. Theory of Computing*, 1998, pp. 614–623.
[22] K.-I. Lin, H. V. Jagadish, and C. Faloutsos, "The TV-tree: An index structure for high-dimensional data," *VLDB J.*, vol. 3, no. 4, pp. 517–542, 1994.
[23] N. Megiddo and U. Shaft, "Efficient nearest neighbor indexing based on a collection of space-filling curves," IBM Almaden Research Center, San Jose, CA, Tech. Rep. RJ 10093, 1997.
[24] M. Miyahara and Y. Yoshida, "Mathematical transform of (R,G,B) color data to Munsell (H, V, C) color data," *Proc. SPIE, Vis. Commun. Image Process.*, vol. 1001, pp. 650–657, 1992.
[25] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin, "The QBIC project: Querying images by content using color, texture and shape," in *Proc. SPIE Conf. Storage and Retrieval for Image and Video Databases II*, 1993, pp. 173–187.
[26] D. Ponceleon, S. Srinivasan, A. Amir, D. Petkovic, and D. Diklic, "Key to effective video retrieval: Effective cataloging and browsing," in *Proc. ACM Multimedia Conf.*, 1998, pp. 99–107.
[27] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest neighbor queries," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 1995, pp. 71–79.
[28] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York: McGraw-Hill, 1983.
[29] J. Shepherd, X. Zhu, and N. Megiddo, "A fast indexing method for multidimensional nearest neighbor search," in *Proc. IS&T/SPIE Conf. Storage and Retrieval for Image and Video Databases VII*, 1999, pp. 350–355.
[30] G. Strang, *Linear Algebra and its Applications*, 2nd ed. New York: Academic, 1980.
[31] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proc. 24th Int. Conf. VLDB*, 1998, pp. 194–205.
[32] P. Zezula, P. Savino, G. Amato, and F. Rabitti, "Approximate similarity retrieval with M-trees," *VLDB J.*, vol. 7, no. 4, pp. 294–307, 1998.

Guang-Ho Cha received the Ph.D. degree in computer engineering from the Korea Advanced Institute of Science and Technology (KAIST), Taejon, South Korea in 1997.

From 1999 to 2000, he was a Visiting Scientist at the IBM Almaden Research Center, San Jose, CA. He is currently an Assistant Professor of Multimedia Engineering, Tongmyong University of Information Technology, Pusan, South Korea. His research interests include content-based image/video indexing and retrieval, XML/semistructured databases, and distance learning.



Xiaoming (Allan) Zhu received the B.S. degree from Peking University, Beijing, China, in 1982, and the M.S. and Ph.D. degrees in geophysics in 1985 and 1989, respectively, from the University of California, Los Angeles, where he developed scientific (data mining) software to identify inherent relations among observations from space probes and ground-based instruments.

He has more than ten years of software development experience in both academic and industry. He is currently with elance.com, Sunnyvale, CA. Before joing elance.com, he was with IBM Almaden Research Center, San Jose, CA, where he worked on content-based image retrieval, digital video processing, pervasive computing, etc. As a Project Leader at IBM, he worked with development teams both inside and outside IBM to push the QBIC image searching technology into various products, such as the DB2 image extender. Before joining IBM, he worked at the University of California, San Francisco, where he built an infrastructure to manage and display radiological images through a central archiving system and to develop image processing tools.

Dragutin Petkovic (S'82–M'83–SM'87–F'99) received the Ph.D. degree in biomedical imaging and computer vision from the University of California, Irvine, in 1983.

He was with IBM Almaden Research Center, San Jose, CA, until 2000. His expertise ranges from machine vision, pattern recognition, and content-based retrieval to internet media. He was involved in several major projects at IBM, among them QBIC and large multimedia databases. In 2000, he joined Dotcast, a start-up in datacasting over the analog TV and held the position of Vice President of SW Development. In March 2001, he joined VMware, Palo Alto, CA, as Senior Director of R&D. He has over 40 refereed publications and five issued patents. He is a reviewer for several major technical journals.

Chin-Wan Chung received the Ph.D. degree from the University of Michigan, Ann Arbor, in 1983.

Since 1993, he has been a Professor in the Department of Computer Science at the Korea Advanced Institute of Science and Technology (KAIST), Taejon, South Korea. Prior to joining KAIST, he was a Senior Research Scientist and a Staff Research Scientist in the Computer Science Department at General Motors Research Laboratories (GMR). While at GMR, he developed DATAPLEX, a heterogeneous distributed database management system integrating relational databases and hierarchical databases. At KAIST, he has developed COSMIC, a content-based multimedia retrieval system using domain knowledge and visual information. He has published extensively in international journals and conferences. His current research interests include multimedia databases, XML, and spatio-temporal databases.

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
l1 near5 l5	2

Database:
- US Patents Full-Text Database
- US Pre-Grant Publication Full-Text Database
- JPO Abstracts Database
- EPO Abstracts Database
- Derwent World Patents Index
- IBM Technical Disclosure Bulletins

Search: L9

Refine Search

Recall Text | Clear

Search History

DATE: Monday, April 01, 2002 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=TDBD; PLUR=YES; OP=OR			
L9	l1 near5 l5	2	L9
L8	l1 and l4	0	L8
L7	l1 and l2	0	L7
L6	vector$	1938	L6
L5	query$ or queries or search$	3502	L5
L4	(partition$ or divid$ or segment$ or break$ or separat$ or split$) near5 (space$ or dataspace$) near5 (cell$ or field$ or compartment$ or block$ or segment$ or section$ or piece$ or column$ or row$)	93	L4
L3	(multidimensional or n or dimensional or data or vector) adj2 space$ or dataspace$	313	L3
L2	(local$ or polar) near3 coordinate$	39	L2
L1	("near" or nearest) adj neighbor$ or knn or (bilinear or bi adj linear) adj interpolation or single adj linkage$ or (best adj match$) near3 (search$ or quer$) or similarity adj join or post adj office adj problem or closest adj point	73	L1

END OF SEARCH HISTORY

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 2 of 2 returned.

☐ 1. Document ID: NN9501415

L9: Entry 1 of 2 File: TDBD Jan 1, 1995

DOCUMENT-IDENTIFIER: NN9501415
TITLE: Fast Heuristic for the Traveling Salesman Problem

Disclosure Text (1):
This document contains drawings, formulas, and/or symbols that will not appear on line. Request hardcopy from ITIRC for complete article. Disclosed is an approximation algorithm for the Traveling Salesman Problem (TSP) which has a vast variety of applications. The disclosed algorithm is a tour construction heuristics that builds a tour from scratch. It uses k-d tree, a data structure that enables fast near-neighbor queries in geometric space. Also it utilizes the van der Corput sequence, one of low-discrepancy sequences that attains the lower bound of discrepancy. The heuristic proceeds as follows. First, it picks up a set of representative points out of all data points so that they well reflect the overall distribution of data points. These representative points are chosen from internal nodes of a k-d tree, which is constructed for later use in near-neighbor queries, in order to distribute them uniformly in the input data region. The heuristic uses the addition method, and constructs a subtour from these representative points. When the addition method is applied, representative points are picked up in the order of van der Corput sequence so that chosen points distribute as uniformly as possible at any stage of subtour construction. The heuristic then grows the subtour thus obtained by applying the addition method for the rest of points. More concrete description of the heuristic is depicted in Fig. 1. Fig. 2 roughly shows how the heuristic works for an instance of size 1127 (a). Figure 2b is a set of representative points, and Fig. 2c is the subtour consisting only of these points. The final tour is shown in Fig. 2d.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc

☐ 2. Document ID: NN9207303

L9: Entry 2 of 2 File: TDBD Jul 1, 1992

DOCUMENT-IDENTIFIER: NN9207303
TITLE: Display of Related Data in Two Windows.

Disclosure Text (1):
- This article describes a method for showing the relationship between data displayed in two separate windows. The problem was that a single window could not accommodate all the relevant and interrelated data required to configure a particular feature. - The goal of this release of Communications Manager was to improve the usability of the configuration dialog. - This release of Communications Manager was the first release that would contain the SNA Distribution System or (SNA/DS) feature. SNA/DS is a store and forward document (mail) distribution system. In order to store and forward documents, SNA/DS maintains a directory and a routing

table. The directory is a list of remote users and their locations. The routing table maps mail for a remote location to the appropriate outgoing queue depending on the priority, security required, etc., for the mail. - The data is related in the following way. The directory looks something like: User Name Location (DGN.DEN) (RGN.REN) account.bob ausnet.ausvm1 account.lucy ausnet.ausvm1 The routing table contains a list of remote locations and the outgoing queues that service them. It looks something like: Location Priority(Etc.) Queue (RGN.REN) ausnet.ausvm1 low CentralProcessing ausnet.ausvm1 high DirectT1 When SNA/DS receives something to forward, it looks up the user's name in the directory and finds the user's location. It then looks up the location in the routing table and finds the appropriate queue to put it on based on the priority and other things that describe the document. - The SNA/DS routing table is very flexible in that the user can use wildcards to put mail for several different locations onto one queue easily. For example the user can create an entry that will work for all locations. Location Priority(Etc.) Queue (RGN.REN) *.* low CentralProcessing This entry will put mail for any location on the CentralProcessing queue. (SNA/DS does a best-match search. It will put high priority mail on the CentralProcessing queue if that is the only one available.) The wildcarding is more flexible than the above example. Users can configure routing entries that look like: Location Priority(Etc.) Queue (RGN.REN) ausnet.ausvm1 low,not secure CentralProcessing ausnet.aus* low,secure ausnet.* high,secure In this case, a document for account.joe at ausnet.ausvm1 might be put on any of the above queues, depending on what kind of data it is. - The SNA/DS feature was already supported on larger machines where a system administrator could be assumed. This was the first time SNA/DS would be supported on a PC. Familiarity with SNA/DS and even the existence of a system administrator could not be assumed. - Previous SNA/DS configurations reflected the internal structure of SNA/DS, e.g., there was a window for the directory and a window for the routing table. - Important user tasks, however, require that the data in the tables be viewed together. Users need to see if there is a route configured for sending mail to a user. Users also need to see what routes are available for sending data to a particular user. (This is important so that the user does not have to guess the SNA/DS lookup algorithm.) Previous implementations did not provide a visual correlation between the directory and routing tables. Users performed the visual lookup themselves. - This release, the first on a Personal Computer, presented the configuration data with its logical relationship, not its internal relationship. Hence, multiple tables might be reflected or modified based on the changes to one window. - The relationship between the SNA/DS directory and routing table can be laid out in a very logical manner on paper, but not easily on a computer screen because of the size restrictions. All of this data is pertinent and needs to be viewed together. The number of entries to be configured cannot be predicted. Hence, scrolling left and right or up and down was not appropriate. What was needed was a clear method of expressing the relationship between directory entries and routing table entries on a single screen. - The directory and routing table are presented in two windows. The two windows do not fit side by side. The last column of the first window is the same as the first column in the second window. The association between the data in the two windows is maintained as the user accesses either window. - The solution is to provide the user a means of showing the relationship between the two windows. For this particular product, the user needed to map the users (DGN.DEN) to the routes going to their locations (RGN.REN). - The DGN.DEN is the key entry in the User and Agent Directory Window. The RGN.REN is the last entry for each DGN.DEN in the User and Agent Window and is the first entry in the Routing window. To see (or not see) the relationship, the user selected SHOW RELATIONSHIP from the VIEW pulldown of either window. The following describes the specific interaction between the two windows. The windows themselves are shown in Figs. 1 and 2. - View - Show Relationship synchronizes the User and Agent Directory and Routing dialog windows. - The two dialog windows interact in the following way: o A row is selected in the User and Agent Directory window. o The available routes for the selected row in the User and Agent Directory are marked in the Routing window with a '>' symbol in column one (1). o The first available route, determined by SNA/DS lookup order, in the Routing window is selected. o The cursor is placed on the selected row of the active window. o The active window is the window from which the option View - Show Relationship was chosen. - The marked rows of the Routing window represent the various paths that a document could take from this workstation to a remote user at the destination RGN.REN. - If the User and Agent Directory window is active, then the available routes that are marked in the Routing window change as the cursor moves through the User and Agent Directory entries. - If the Routing window is active, then the User and Agent Directory window remains unchanged, i.e., the same row remains selected, while the cursor moves through the marked Routing rows. - The View-Show Relationship selection in the pull-down acts as a toggle to turn on/off the synchronization

between the two windows. The cursor only moves through the marked routes while the toggle is turned on. The cursor moves through all routes when the toggle is turned off. The toggle is changed from the VIEW pulldown of either window. A check mark in front of SHOW RELATIONSHIP indicates that the toggle is on. - The wildcard routing entry, RGN.REN set to *.*, is always marked as an available route, if such an entry exists.

Terms	Documents
l1 near5 l5	2